

[00176] APPENDIX A:

module BLE

```
{
    interface AgCredentialsManager;
    interface AgCredentials;
    interface AgInstance;
    interface DomainServiceFactory;
    interface DomainService;

    //-----
    // DomainServiceFactory
    //-----
    interface DomainServiceFactory
    {
        DomainService createDomainService(in string domain);
        // create domain service for the domain
    };

    interface DomainService
    {
        AgCredentialsManager createCredentialsMgr(in string application);
        // create credentials manager for the application
    };

    // This struct is set by the rule(s) evaluated and may contain
    // output information, both from the rule and global
    struct EvaluationResult {
        string ruleID;
        string privilege;
        string objectName;
        TRUTH_VALUE_ENUM decision;
        NVPairList data;
    };

    typedef sequence<EvaluationResult> EvaluationResultList;

    struct AccessElement
    {
        string privilege;
        string objectName;
        BLE::NVPairList inAttrList;
        boolean findAllFacts;
        BLE::EvaluationResultList results;
        TRUTH_VALUE_ENUM accessAllowed;
    };

    typedef sequence<AccessElement> AccessList;

    typedef sequence<string> PrivilegeList;

    typedef sequence<string> ObjectList;

    typedef sequence<string> RoleList;
```

```

interface AgCredentials : Common::WBOobject
{
    string getDomainName();
    // get domain name

    string getLocationName();
    // get location name

    string getApplicationName();
    // get application name

    string getUserId();
    // get userid

    TRUTH_VALUE_ENUM accessAllowed(in BLE::NVPairList inAttrs,
                                   in string privilege,
                                   in string objectName,
                                   in boolean findAllFacts,
                                   out BLE::EvaluationResultList
results)
        raises(Common::BadParameterException,
                BLE::CredvarException,
                BLE::InternalException,
                BLE::LogicException,
                BLE::InvalidUserException);
    // Solve policy. "inAttrs" is a list of input dynamic attributes
    // for the request. "results" is (possibly empty) list of
    // EvaluationResult data structures set by the BLE engine.
    // FindAllFacts, when set to true continues rules evaluation
    // after first deny or grant is found, to allow all
    // potentially firing rules to report any output attributes
    // as set by the administrator of the policy - it
    // should not be used if output attributes are not used
    // as it slows down evaluation considerably

    void bulkAccessAllowed(inout BLE::AccessList accessList)
        raises(Common::BadParameterException,
                BLE::CredvarException,
                BLE::InternalException,
                BLE::LogicException,
                BLE::InvalidUserException);
    // solve policy in bulk. All evaluation requests in accessList
    // will be processed at the same time.

    BLE::PrivilegeList queryPrivileges(in string objectName,
                                       in boolean includedeny,
                                       in BLE::NVPairList inAttrs,
                                       in boolean findAllFacts,
                                       out BLE::EvaluationResultList
results)
        raises(Common::BadParameterException,
                BLE::CredvarException,
                BLE::InternalException,
                BLE::LogicException,
                BLE::InvalidUserException);
    // query privileges on the object. if includedeny is true

```

```

        // auth eval will be computed on every priv-obj-subj
combination
        // and all grants will be returned in privileges list.
        // if includedeny is false, no auth eval is done.
        // note that query result includes privilege propagation
        // on the object tree, so you do not know if privilege is
        // written directly on the object or is inherited

        BLE::ObjectList queryObjects(in string privilege,
                                     in string clippingNode,
                                     in boolean includedeny,
                                     in BLE::NVPairList inAttrs,
                                     in boolean findAllFacts,
                                     out BLE::EvaluationResultList
outAttrs)
        raises(Common::BadParameterException,
               BLE::CredvarException,
               BLE::InternalException,
               BLE::LogicException,
               BLE::InvalidUserException);
        // Query objects below clipping node for the specified
privilege.
        // If includedeny is true AccessAllowed will be computed on
every
        // priv-obj-subj combination and all objects below clipping
node
        // will be returned in objects list. If includedeny is false
        // no evaluation is done.

        boolean queryPerformance(out double averageQueryLatency,
                                out double averageQueryLoad);

        =
        // This call returns average AccessAllowed evaluation time
        // in seconds and average load on the engine - it is
        // available only if auditing is enabled - will return false
        // otherwise.
    };

    interface AgCredentialsManager
    {
        string getDomainName();
        // get domain name

        string getLocationName();
        // get location name

        string getApplicationName();
        // get application name

        AgCredentials findCredentials(in string userid)
        raises(Common::BadParameterException,
               BLE::InvalidUserException,
               BLE::InternalException,
               BLE::LogicException);
        // Find credentials for the userid. Either new or existing
        // credentials object can be returned.

```

```

AgCredentials findCredentialsWithRoles(in string userid,
                                      in BLE::RoleList roles)
    raises(Common::BadParameterException,
           BLE::InvalidUserException,
           BLE::InternalException,
           BLE::LogicException);
    // Find credentials for the userid and roles.
    // Either new or existing
    // credentials object can be returned.

};

//-----
// AgInstance
//-----

struct BindingDelta
{
    string action; // add
    string agname;
    string application;
};

typedef sequence<BindingDelta> BindingDeltaSeq;

struct DirectoryDelta
{
    string action; // del, ren
    string directory;
    string newDirectory;
};

typedef sequence<DirectoryDelta> DirectoryDeltaSeq;

struct UserDelta
{
    string action; // del, ren, add
    string user;
    string newUser;
};

typedef sequence<UserDelta> UserDeltaSeq;

struct RoleDelta
{
    string action; // del, ren, add
    string role;
    string newRole;
};

typedef sequence<RoleDelta> RoleDeltaSeq;

struct RoleMemberDelta
{
    string action; // del, add
    string role;
    string member;
};

```

```

};

typedef sequence<RoleMemberDelta> RoleMemberDeltaSeq;

struct GlobalUserMappingDelta
{
    string action; // del, add
    string globalUser;
    string localUser;
};

typedef sequence<GlobalUserMappingDelta> GlobalUserMappingDeltaSeq;

struct GlobalRoleMappingDelta
{
    string action; //del, add
    string globalRole;
    string localRole;
};

typedef sequence<GlobalRoleMappingDelta> GlobalRoleMappingDeltaSeq;

struct GlobalSubjectDelta
{
    string action; // ren, del
    string globalSubject;
    string newGlobalSubject;
    string mappedDirectory;
};

typedef sequence<GlobalSubjectDelta> GlobalSubjectDeltaSeq;

struct SubjectAttributeDelta
{
    string action; // add, del
    string subject;
    string attr;
    string value;
    string type; // single: S, list: L
};

typedef sequence<SubjectAttributeDelta> SubjectAttributeDeltaSeq;

struct ObjectAttributeDelta
{
    string action; // add, del
    string objectName;
    string attr;
    string value;
    string type; // single: S, list: L
};

typedef sequence<ObjectAttributeDelta> ObjectAttributeDeltaSeq;

struct LogicalNamesDelta
{
    string action; // del, mod, add

```

```

        string objectName;
        string logicalName;
    };

    typedef sequence<LogicalNamesDelta> LogicalNamesDeltaSeq;

    struct ObjectDelta
    {
        string action; // del, ren, add
        string objectName;
        string newObjectName;
        string type; // A, 0 (this is for object)
    };

    typedef sequence<ObjectDelta> ObjectDeltaSeq;

    struct DeclDelta
    {
        string action; // del, mod, add (ren = del -> add)
        string text;
    };

    typedef sequence<DeclDelta> DeclDeltaSeq;

    struct RuleDelta
    {
        string action; // del, add
        string rule; // posid:rid:text (add); posid (del)
    };

    typedef sequence<RuleDelta> RuleDeltaSeq;

    interface AgInstance //: SG::ServerGroupMember //: Common::WBOBJECT
    {
        string getAgLocalName();
        // get ag instance name

        string getDomainName();
        // get domain name

        string getLocationName();
        // get location name

        AgCredentialsManager getAgCredentialsManager(in string
application)
            raises(Common::BadParameterException);
        //returns a CredsMgr for a given application
        //raises badparm if application is not guarded by this AG

        void startPolicyUpdate()
            raises(Common::RuntimeException);
        // start policy update

        void saveBindingDelta(in BindingDeltaSeq seq, in boolean more)
            raises(Common::RuntimeException);
        // save binding delta

```

```

more) void saveDirectoryDelta(in DirectoryDeltaSeq seq, in boolean
      raises(Common::RuntimeException);
      // save directory delta

void saveUserDelta(in UserDeltaSeq seq, in boolean more)
      raises(Common::RuntimeException);
      // save user delta

void saveRoleDelta(in RoleDeltaSeq seq, in boolean more)
      raises(Common::RuntimeException);
      // save role delta

more) void saveRoleMemberDelta(in RoleMemberDeltaSeq seq, in boolean
      raises(Common::RuntimeException);
      // save role membership delta

seq, void saveGlobalUserMappingDelta(in GlobalUserMappingDeltaSeq
      in boolean more)
      raises(Common::RuntimeException);
      // save global user mapping delta

seq, void saveGlobalRoleMappingDelta(in GlobalRoleMappingDeltaSeq
      in boolean more)
      raises(Common::RuntimeException);
      // save global role mapping delta

void saveGlobalSubjectDelta(in GlobalSubjectDeltaSeq seq,
      in boolean more)
      raises(Common::RuntimeException);
      // save global subject delta

void saveSubjectAttributeDelta(in SubjectAttributeDeltaSeq seq,
      in boolean more)
      raises(Common::RuntimeException);
      // save user attribute delta

void saveLogicalNamesDelta(in LogicalNamesDeltaSeq seq,
      in boolean more)
      raises(Common::RuntimeException);
      // save logical names delta

void saveObjectDelta(in ObjectDeltaSeq seq, in boolean more)
      raises(Common::RuntimeException);
      // save object tree delta

void saveObjectAttributeDelta(in ObjectAttributeDeltaSeq seq,
      in boolean more)
      raises(Common::RuntimeException);
      // save object attribute delta

void saveDeclDelta(in DeclDeltaSeq seq, in boolean more)
      raises(Common::RuntimeException);
      // save decl delta

```

```

void saveRuleDelta(in RuleDeltaSeq seq, in boolean more)
    raises(Common::RuntimeException);
// save rule delta

string prepareToCommit(in long policyid, in boolean flush)
    raises(InvalidDataException,
        Common::RuntimeException);
// prepare to commit policy update, return policy hash
// input is new policy id and a flush flag, that instructs
// app guard to flush it's current policy

void commit();
// commit policy update

void rollback();
// rollback policy update

oneway void invitationToRegister();
// register with policy distributor

long getPolicyId();
// get app guard policy id

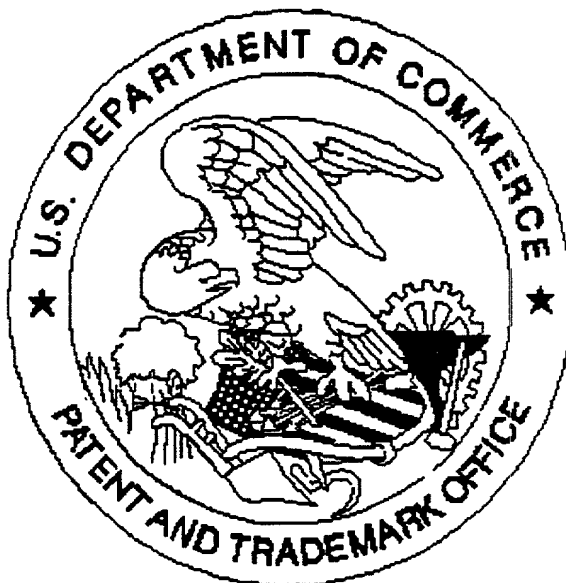
double getProcessingRate();
// returns current moving average of the number of requests
processed
// per second

    };
[00177] };

[00178]

```


United States Patent & Trademark Office
Office of Initial Patent Examination -- Scanning Division



Application deficiencies found during scanning:

☐ Page(s) _____ of _____ were not present
for scanning. (Document title)

☐ Page(s) _____ of _____ were not present
for scanning. (Document title)

☒ **Scanned copy is best available.**

pages numbered 51 to
58 of specification are
appendix.